**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ……………………………………………..…………………..4

АНАЛИЗ ПОСТАНОВКИ ЗАДАЧИ……………………..……………….5

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ…………………………………….…….…...7

ПРОЕКТИРОВАНИЕ ПРОГРАММЫ………………….…….….……….9

РЕАЛИЗАЦИЯ ПРОГРАММЫ…………………………….…….………11

ТЕСТИРОВАНИЕ ПРОГРАММЫ……………………………….……....16

ЗАКЛЮЧЕНИЕ………………………………………....….……...….…....21

СПИСОК ЛИТЕРАТУРЫ……………………………………..……......…22

ПРИЛОЖЕНИЕ А……………………………………………..…….….….23

ПРИЛОЖЕНИЕ Б………………………………………….…...……...…...25

**ВВЕДЕНИЕ**

Язык С++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Синтаксис С++ унаследован от языка С. С++ поддерживает такие парадигмы программирования, как процедурное программирование, объектно- ориентированное программирование, обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространенные контейнеры и алгоритмы, ввод - вывод, регулярные выражения, поддержку многопоточности и другие возможности. С++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

Часто в процессе работы с данными невозможно определить, сколько памяти потребуется для их хранения, память следует распределять во время выполнения программы по мере необходимости отдельными блоками. Блоки связываются друг с другом с помощью указателей. Такой способ организации данных называется динамической структурой данных, поскольку она размещается в динамической памяти и ее размер изменяется во время выполнения программы.

Из динамических структур в данной работе используются линейные списки. Динамическая структура, в отличие от массива или записи, может занимать несмежные участки оперативной памяти. Динамические структуры широко применяют и для более эффективной работы с данными, размер которых известен, особенно для решения задач сортировки.

Целью курсовой работы является создание консольного приложения базы данных. Обработка данных должна производиться в динамическом списке связанного хранения.

Большинство приложений, которые предназначены для выполнения какой-либо полезной работы, тем или иным образом используют структурированную информацию. Так называемые упорядоченные данные. При компьютерной обработке информации данные, упорядоченные каким-либо образом, принято хранить в базах данных.

База данных (БД)- это программа, которая позволяет хранить и обрабатывать информацию в структурированном виде. БД это отдельная независимая программа, которая не входит в состав языка программирования. В базе данных можно сохранять любую информацию, чтобы позже получать к ней доступ. Это особые файлы, использование которых вместе со специальными программными средствами позволяет пользователю, как просматривать необходимую информацию, так и манипулировать ею: добавлять, изменять, копировать, удалять или сортировать данные.

**1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧИ**

Цель создания программы: необходимо создать консольное приложение, позволяющее хранить данные об абитуриентах, поступающих в университет.

Назначение программы:

Программа должна выдать на печать следующую информацию:

* фамилию, имя и отчество абитуриента;
* дату рождения абитуриента;
* оценки по трем предметам, полученные при сдаче вступительных экзаменов;
* общий балл, полученный при сложении оценок;
* шифр выбранной специальности.

В программе должны быть реализованы следующие возможности:

* добавлять элементы в двусвязный список и записывать введенные данные в текстовый файл;
* считывать из файла данные и добавлять их в двусвязный список;
* корректировать данные в списке;
* выводить на экран список данные об абитуриентах (ФИО, дата рождения, полученные оценки по трем предметам, общий балл и шифр выбранной специальности);
* сортировать данные по фамилии абитуриента, по дате рождения и по общему баллу, как по возрастанию, так и по убыванию;
* фильтровать список абитуриентов по выбранной специальности;
* печатать списки поступивших в университет студентов, как по отдельным специальностям, так и общим списком.

В процессе разработки программы понадобится использовать следующие функции из стандартных библиотек языка С/С++:

* Функции system(«cls») и system(«pause») из библиотеки Windows.h. Функция system(«cls») служит для очистки консоли, функция system («pause»)- для задержки консольного окна.
* Для работы с файлами необходимо подключить заголовочный файл fstream. Для того чтобы открыть файл для чтения, нужно создать объект класса ifstream, указав имя файла в качестве параметра. Для того чтобы создать поток для записи в файл, нужно создать объект класса ofstream, также указав в качестве параметра имя файла. Затем, с помощью оператора вставки (<<) и оператора извлечения (>>) можно записывать данные в файл или считывать содержимое из файла. После работы с файлом необходимо использовать функцию close() , для закрытия файла. Функция close() параметров не имеет.
* Для разграничения границ таблицы и создания ровных столбцов, необходимо подключить библиотеку iomanip и использовать манипулятор setw(), который устанавливает ширину поля, а также манипулятор setfill() из этой же библиотеки, который устанавливает символ заполнения пустого поля таблицы. Манипулятор endl выводит символ новой строки и передает в поток всех данных из буфера.
* Функция swap() предназначена для обмена двух значений. Она принимает в качестве параметров две переменные, которые необходимо переставить местами.
* Функция strcmp() работает со строками. В качестве параметров, принимаются две строки, которые сравниваются между собой. Для работы со строками необходимо подключить заголовочный файл string.

Разработанная программа имеет ограничения в возможностях. Для упрощения программного кода, в процессе разработки при выводе данных на экран используются некоторые ограничения для вводимых символов (таблица 1.1)

**Таблица 1.1** ограничения в программе на вводимые данные

|  |  |
| --- | --- |
| **Вводимые данные** | **Ограничения** |
| Фамилия, имя, отчество | строки по 20 символов |
| Оценки | количество предметов- 3 |
| Оценки | целые числа от 1 до 10 |
| Формат вывода даты | дд.мм.гггг |
| Шифр предмета | строка 2 символа |

**2.ТЕОРИТИТИЧЕСКАЯ ЧАСТЬ**

Целью большинства программ является решение какой-либо задачи путем различных преобразований исходных данных. Для этого необходимо:

* ввести данные в программу;
* установить места для хранения информации (выделить область памяти);
* реализовать инструкции обработки информации;
* вывести информацию из программы;
* организовать выполнение некоторых инструкций лишь тогда, когда какое- либо условие, или набор условий дает требуемое значение (например, истинное или ложное);
* реализовать возможность повторения инструкции или группы инструкций некоторое число раз;
* выделить группы инструкций, которые можно вызывать и выполнять в различных частях программы (просто указав имя этой группы).

В данных пунктах определены семь основных элементов программирования: ввод, типы данных, операторы, вывод, проверка условий, циклы и подпрограммы. Язык С++ наряду с этими обязательными компонентами имеет и некоторые дополнительные конструкции. Рассмотрим содержательную характеристику каждого из рассматриваемых элементов.

**Ввод** - чтение данных с клавиатуры, магнитного диска, порта ввода- вывода, или из другой программы.

**Данные** - могут быть простыми либо иметь структурированный тип. В программах на С++ ими являются числа, текстовая информация и адреса, указывающие места размещения других данных.

**Операторы** - присваивают значения переменным, реализуют переходы между различными инструкциями программы, вызывают подпрограммы и т.п. Их важный подкласс составляют операторы перехода и операторы цикла.

**Вывод** - отображение информации на экране дисплея, ее запись на магнитный диск, передача в порт ввода - вывода или в другую программу.

**Операторы условного перехода** - позволяют выбирать различные маршруты обработки и преобразования данных в зависимости от истинности или ложности некоторых заданных условий.

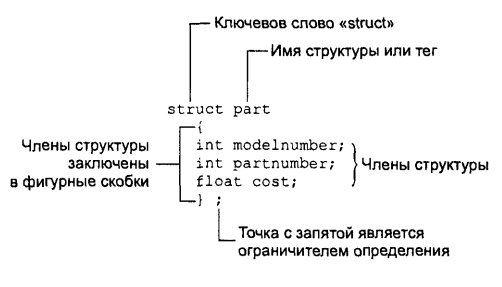
**Операторы цикла** - реализуют выполнение одной или нескольких инструкций заданное число раз либо до тех пор, пока не удовлетворится некоторое условие.

**Подпрограммы** - отдельно именуемые наборы инструкций, которые могут быть вызваны (сколько угодно раз) и выполнены в любом месте программы. В языке С++ подпрограммы именуют функциями.

**Структура** – это группа связанных переменных. В С++ структура представляет собой коллекцию объединенных общим именем переменных, которая обеспечивае6т удобное средство хранения родственных данных в одном месте. Структуры - это совокупные типы данных, поскольку они состоят из нескольких различных, но логически связанных переменных.

Прежде, чем будет создан объект структуры, должен быть определен ее формат. Это делается посредством объявления структуры. Объявление структуры позволяет понять, переменные, какого типа она содержит. Переменные, составляющие структуру, называются ее *членами*. Члены структуры также называют элементами или полями.

**Член структуры** – это переменная, которая является частью структуры. В общем случае, все члены структуры должны быть логически связаны друг с другом.



**Рисунок 2.1** Шаблон структуры

**Связный список** - структура, элементы которой имеют один и тот же формат и связаны друг с другом с помощью указателей, хранящихся в самих элементах. В линейном двусвязном списке продвижение возможно в любом из двух направлений по цепочке элементов. Каждый элемент двусвязного списка содержит два указателя: указатель на следующий элемент, или прямой указатель, и указатель на предыдущий элемент, или обратный указатель.

Логическая структура элемента линейного двусвязного списка:

Данные, или указатель на данные, указатель на следующий элемент списка, указатель на предыдущий элемент списка. Схема двусвязного списка предоставлена на рисунке 2.2.

Данные

Следующий

Предыдущий

Данные

Следующий

Предыдущий

Данные

Следующий

Предыдущий

NULL NULL

**Рисунок 2.2** Двусвязный список

Для достижения поставленной цели в курсовой работе необходимо решить следующие задачи:

* Описать постановку задачи, методы решения, используемые средства (библиотеки, функции и операторы);
* разработать схему алгоритма программы;
* описать созданные пользовательские функции;
* разработать руководство пользователя;
* описать тестовый пример работы программы;

**3 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ**

При открытии программы, пользователь увидит, меню с пунктами, содержащими предлагаемые к выполнению действия:

* Выход из программы.
* Добавление данных.
* Печать данных.
* Изменение данных.
* Удаление данных.
* Фильтр данных.
* Сортировка данных.
* Сохранение данных.
* Вывод на экран списка поступивших в университет студентов.
* Поиск студента по фамилии.

При нажатии пользователем клавиши «1» на экране появится надпись, предлагающая пользователю способ ввода данных: ввести вручную (нажать клавишу «1») или добавить данные из файла, для чего потребуется нажать любую клавишу.

При нажатии клавиши «2», пользователь имеет возможность вывести на экран список абитуриентов.

Для того, чтобы изменить данные, необходимо нажать клавишу «3». Программа запросит порядковый номер студента, чьи данные необходимо изменить. После этого, следуя указаниям на экране, пользователь сможет изменить данные.

Для удаления данных необходимо нажать клавишу «4». Программа запросит порядковый номер удаляемого элемента. Если элемент с таким номером существует, то данные будут удалены. В противном случае программа выведет сообщение об ошибке.

Чтобы отфильтровать данные, необходимо выбрать пункт с номером «5», нажав соответствующую клавишу. Пользователь увидит на экране меню, с помощью которого сможет выбрать, по каким критериям необходимо фильтровать данные.

При выборе пункта с номером «6» в главном меню, предоставляется возможность сортировки данных. Программа выполняет сортировку списка по дате рождения, по сумме баллов и по фамилии. Для того, чтобы сделать выбор нужно нажать клавишу «1», «2» или «3». После того, как пользователь сделает свой выбор, ему нужно будет выбрать в каком порядке необходимо сортировать данные : по возрастанию или по убыванию, после чего на экране появится таблица с отсортированными данными.

Для сохранения данных в файл, пользователю потребуется ввести цифру «7». Программа предложит ввести название файла, в который необходимо сохранить данные. Название файла вводится без кавычек, указав расширение файла.

Чтобы вывести на экран списки зачисленных в университет студентов, необходимо выбрать пункт номер «8», нажав соответствующую клавишу. Программа предложит пользователю выбрать показать список зачисленных студентов на определенную специальность или же список всех зачисленных на первый курс студентов. Выбрав интересующий пункт меню, пользователь, получит интересующую его информацию.

Поиск абитуриента по фамилии реализован в пункте «9» главного меню. Пользователь вводит фамилию, если абитуриент с такой фамилией есть в списке, то на экран выводится информация о данном человеке. В противном случае, на экран выводится информация о том, что поиск не увенчался успехом.

Для завершения работы программы необходимо выбрать пункт «выход из программы», нажав на клавиатуре цифру «0».

**4. РЕАЛИЗАЦИЯ ПРОГРАММЫ**

Для реализации программы были созданы следующие структуры данных:

**Листинг 4.1.** Структура Person:

1. struct Person {
2. char surname[20];
3. char name[20];
4. char patronymic[20];
5. };

**Листинг 4.2.** Структура Data:

1. struct Data {
2. int day;
3. int month;
4. int year;
5. };

**Листинг 4.3.** Структура Grades:

1. struct Grades {
2. char code[2];
3. int subject1;
4. int subject2;
5. int subject3;
6. int sum;
7. };

**Листинг 4.4.** Структура Student:

1. struct Student {
2. int npp;//порядковый номер
3. Person person;
4. Data data;
5. Grades grades;
6. };

Структура tag\_obj, содержащая в себе значение и указатели на предыдущий и следующий элементы.

**Листинг 4.5.** Структура tag\_obj:

1. typedef struct tag\_obj {
2. Student stud;
3. struct tag\_obj \*prev, \*next;
4. }OBJ;

**Листинг 4.6.** Список функций:

1. void InitNode();
2. void InitObj();//инициализация
3. void AddObj();//ввод данных вручную
4. void Print();//печать данных
5. void ReadingObj();//считывание данных из файла
6. void ChangeObj();//изменение данных
7. void DelObj();//удаление элемента
8. void SavingObj();//сохранение данных в файл
9. void SavingCurrent();//сохранение текущих изменений
10. void Sorting\_su();//сортировка по сумме баллов по убыванию
11. void Sorting\_sv();//сортировка (сумма баллов) по возрастанию
12. void Sorting\_du();//сортировка даты по убыванию
13. void Sorting\_dv();//сортировка даты по возрастанию
14. void Sorting\_abc\_V();//сортировка по алфавиту по возрастанию
15. void Sorting\_abc\_U();//сортировка по алфавиту по убыванию
16. void Filter(int a);//фильтр по профессии
17. void FilterSt1(int \_n, int plan);//списки поступивших студентов
18. void Searchname();//поиск по фамилии

Функция Searchname() выполняет поиск студента по фамилии. Пользователь вводит фамилию. При помощи функции из стандартной библиотеки strcmp, которая принимает в качестве параметров две строки, сравнивает между собой введенные пользователем данные с данными в списке. В случае совпадения данных, выводит их на экран. Листинг данной функции имеет вид:

**Листинг 4.7** – выполнение функции Searchname():

1. void Searchname()
2. {
3. OBJ \*obj;
4. obj = head;
5. char n[20];
6. cout << "Введите фамилию ";
7. cin >> n;
8. system("cls");
9. int count = 0;
10. while (obj != NULL) {
11. for (int i = 0; i < quantity; i++) {
12. if (strcmp(n, obj->stud.person.surname) == 0) {
13. count++;
14. if (count == 1) {
15. cout << "+-------------------------------------------------------------------+" << endl;
16. cout << setw(3) << setfill(' ') << "№";
17. cout << left << setw(20) << setfill(' ') << " ФИО абитуриента";
18. cout << left << setw(20) << setfill(' ') << " дата рождения";
19. cout << left << setw(20) << setfill(' ') << " оценки";
20. cout << left << setw(3) << setfill(' ') << " балл";
21. cout << left << setw(3) << setfill(' ') << " шифр " << endl;
22. cout << "+-----------------------------------------------------------------+" << endl;
23. }
24. cout << endl;
25. //№пп
26. cout << right << setw(3) << setfill(' ') << count << " ";
27. //фио
28. cout << left << setw(20) << setfill(' ') << obj->stud.person.surname << endl
29. << " " << left << setw(20) << setfill(' ') << obj->stud.person.name << endl
30. << " " << left << setw(20) << setfill(' ') << obj->stud.person.patronymic;
31. //дата
32. cout << right << setw(2) << setfill('0') << obj->stud.data.day << "."
33. << setw(2) << obj->stud.data.month << "."
34. << setw(4) << obj->stud.data.year << "\t";
35. //оценки
36. cout << setw(2) << setfill(' ') << obj->stud.grades.subject1 << "\t"
37. << setw(2) << setfill(' ') << obj->stud.grades.subject2 << "\t"
38. << setw(2) << setfill(' ') << obj->stud.grades.subject3 << "\t";
39. cout << setw(2) << setfill(' ') << obj->stud.grades.sum << "\t";
40. cout << setw(2) << setfill(' ') << obj->stud.grades.code << endl;
41. cout << "+------------------------------------------------------------------+" << endl;
42. }
43. obj = obj->next;
44. }
45. cout << " По вашему запросу ничего не найдено!" << endl;
46. }
47. }

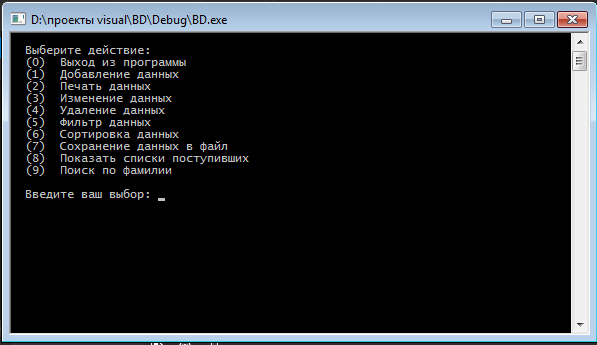
Функция Filter(int n) позволяет осуществить фильтр по шифру специальности. В главном меню пользователь выбирает значение, которое соответствует коду специальности. Это значение принимается функцией в качестве параметра. Указатель устанавливается на начало списка и с помощью цикла while проходит по всему списку, сравнивая данные и переставляя указатель на следующий элемент.

**Листинг 4.8** – выполнение функции Filter(int n):

1. void Filter(int n) {
2. OBJ \*obj;
3. Sorting\_su();
4. obj = head;
5. int count = 0;
6. string cod;
7. if (n == 1) {
8. cout << "+-------------------------------------------------------------------+" << endl;
9. cout << " Специальность 'ф1' " << "\t План приема : " << f1 << endl;
10. cod = «ф1»;
11. }
12. if (n == 2) {
13. cout << "+-------------------------------------------------------------------+" << endl;
14. cout << " Специальность 'ф2' " << "\t План приема : " << f2 << endl;
15. cod = «ф2»;
16. }
17. if (n == 3) {
18. cout << "+------------------------------------------------------------------+" << endl;
19. cout << " Специальность 'ф3' "<< "\t План приема : " << f3 << endl;
20. cod = «ф3»;
21. }
22. while (obj != NULL) {
23. if (obj->stud.grades.code == cod)
24. {
25. count++;
26. if (count == 1) {
27. //формирование столбцов
28. cout << "+-------------------------------------------------------------------+" << endl;
29. cout << setw(3) << setfill(' ') << "№";
30. cout << left << setw(20) << setfill(' ') << " ФИО абитуриента";
31. cout << left << setw(20) << setfill(' ') << " дата рождения";
32. cout << left << setw(20) << setfill(' ') << " оценки";
33. cout << left << setw(3) << setfill(' ') << " балл";
34. cout << left << setw(3) << setfill(' ') << " шифр " << endl;
35. cout << "+------------------------------------------------------------------+" << endl;
36. }
37. if (count > 0) {
38. count++;
39. cout << endl;
40. //№пп
41. cout << right << setw(3) << setfill(' ') << count << " ";
42. //фио
43. cout << left << setw(20) << setfill(' ') << obj->stud.person.surname << endl
44. << " " << left << setw(20) << setfill(' ') << obj->stud.person.name << endl
45. << " " << left << setw(20) << setfill(' ') << obj->stud.person.patronymic;
46. //дата
47. cout << right << setw(2) << setfill('0') << obj->stud.data.day << "."
48. << setw(2) << obj->stud.data.month << "."
49. << setw(4) << obj->stud.data.year << "\t";
50. //оценки
51. cout << setw(2) << setfill(' ') << obj->stud.grades.subject1 << "\t"
52. << setw(2) << setfill(' ') << obj->stud.grades.subject2 << "\t"
53. << setw(2) << setfill(' ') << obj->stud.grades.subject3 << "\t";
54. cout << setw(2) << setfill(' ') << obj->stud.grades.sum << "\t";
55. cout << setw(2) << setfill(' ') << obj->stud.grades.code << endl;
56. cout << "+-----------------------------------------------------------------+" << endl;
57. }
58. }
59. obj = obj->next;
60. }
61. }

**5.ТЕСТИРОВАНИЕ ПРОГРАММЫ**

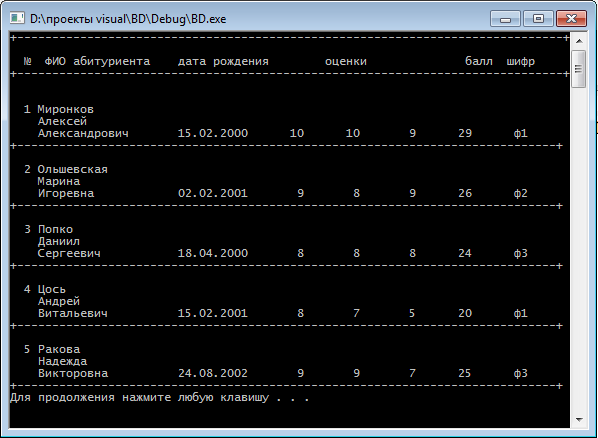
После запуска программы на экране будет отражено меню (рисунок 5.1). Пользователю необходимо ввести цифру, соответствующую выбранным действиям.



**Рисунок 5.1** меню программы

При запуске в программу загружаются данные из файла с названием Input.txt. Текстовый файл содержит данные об актуальном количестве данных в списке, ФИО абитуриентов, дату рождения, оценки по 3 предметам, итоговый балл и шифр выбранной специальности. Данные каждого абитуриента отмечены порядковым номером.

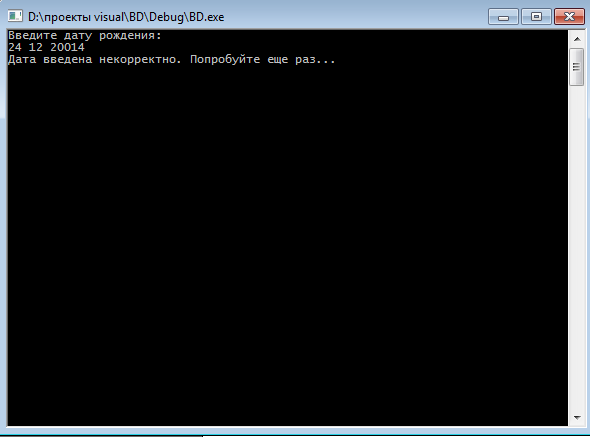
Для вывода списка на экран, необходимо выбрать пункт «Печать данных», нажав для этого цифру 2 (рисунок 5.2)



**Рисунок 5.2** печать списка

Для того, чтобы добавить данные в список, необходимо выбрать пункт «Добавление данных», нажав цифру 1.

На экране появится надпись, уточняющая у пользователя, каким образом он хочет добавить данные: ввести вручную или добавить из файла. Чтобы ввести данные вручную, необходимо ввести цифру 1, ввести данные, запрашиваемые программой: ФИО абитуриента, его дата рождения, оценки и шифр специальности, следуя запросам программы. Если данные введены неверно, программа выведет сообщение об ошибке (рисунок 5.3).

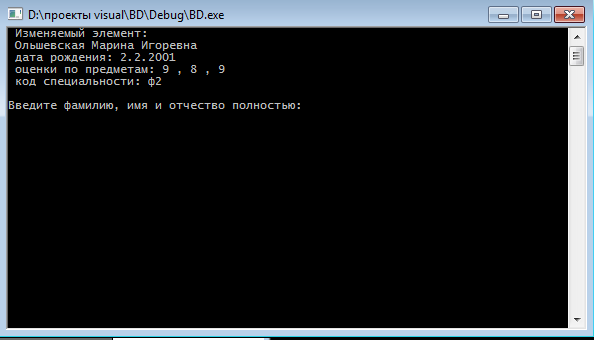


**Рисунок 5.3** отображение ошибки на экране

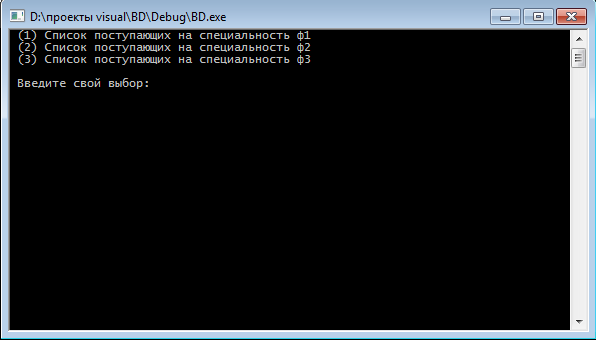
Для изменения данных необходимо выбрать в меню пункт «Изменение данных», нажав на клавишу «3».Программа предложит ввести номер элемента, данные которого необходимо изменить. На экране отобразятся данные изменяемого элемента и предложение ввести новые данные (рисунок 3.4). При успешном вводе данных, программа сообщит, что операция прошла успешно и данные изменены. Для возврата в меню, следует нажать любую клавишу.

Под пунктом «4» - удаление данных. Программа запрашивает номер элемента, который необходимо удалить. Пользователь вводит число и программа сообщает, что данные под этим номером удалены.

Для того чтобы отфильтровать данные, необходимо выбрать пункт под номером «5». Пользователю предлагается выбрать по шифру, какой специальности необходимо отфильтровать данные (рисунок 5.5). Выбрав значение, пользователю на экран выводится список всех абитуриентов, желающих поступить на данную специальность. В самом верху таблицы указана информация, по какому критерию проводился отбор и какой план приема на данную специальность. Данные выводятся на экран отсортированными, поэтому есть возможность предварительно оценить свои шансы на поступление в университет на данную специальность (рисунок 5.6).

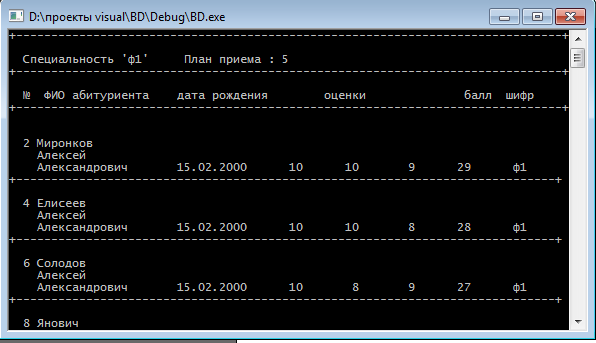


**Рисунок 5.4** изменение данных



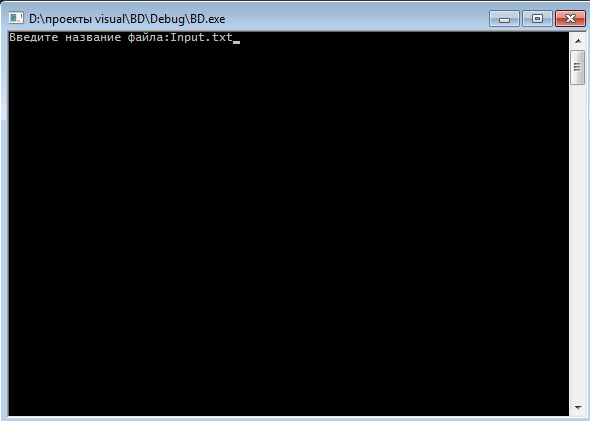
**Рисунок 5.5** фильтр данных

Для сортировки данных потребуется ввести цифру под номером «6». Программа предложит пользователю отсортировать данные по фамилии, дате рождения и по общему баллу. Для того чтобы сделать выбор нужно нажать клавишу 1, 2 или 3. После того как пользователь сделает свой выбор, ему нужно будет выбрать в каком порядке необходимо сортировать данные: по возрастанию или по убыванию, после чего на экране появится таблица с отсортированными данными.



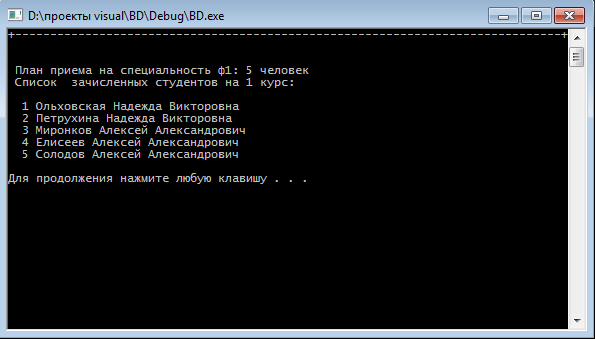
**Рисунок 5.6** отображение на экране таблицы с отфильтрованными данными по шифру «ф1"

Для сохранения данных в файл, пользователю потребуется ввести цифру 7. Программа предложит ввести название файла, в который необходимо сохранить данные. Название файла вводится без кавычек, указав расширение файла (рисунок 5.7) Как будет выглядеть текстовый файл можно увидеть на рисунке 5.8.



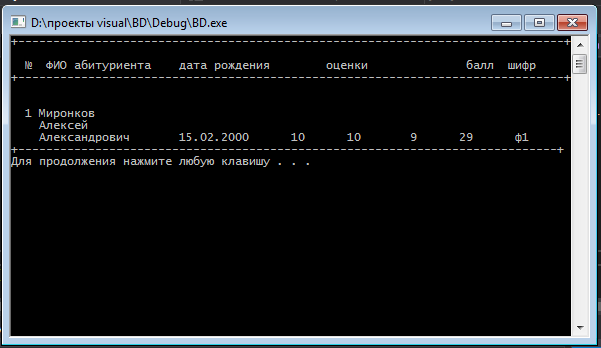
**Рисунок 5.7** сохранение данных в файл

Чтобы вывести на экран списки зачисленных в университет студентов, необходимо выбрать пункт номер 8, нажав соответствующую клавишу. Программа предложит пользователю выбрать показать список зачисленных студентов на определенную специальность или же список всех зачисленных на первый курс студентов. Выбрав интересующий пункт меню, пользователь, получит интересующую его информацию (рисунок 5.9)

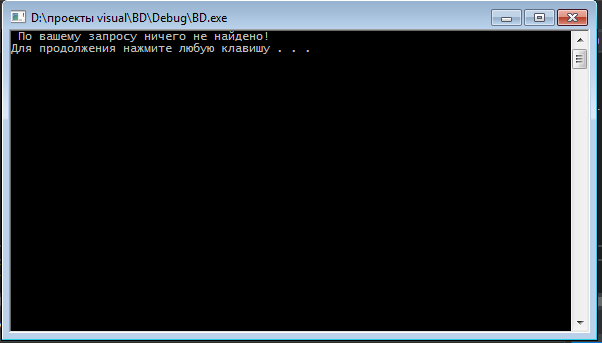


**Рисунок 5.9** список зачисленных студентов

Поиск абитуриента по фамилии реализован в пункте «9» главного меню. Пользователь вводит фамилию, если абитуриент с такой фамилией есть в списке, то на экран выводится информация о данном человеке. В противном случае, на экран выводится информация о том, что поиск не увенчался успехом (рисунки 5.10 и 5.11).



**Рисунок 3.10** поиск успешен



**Рисунок 5.11** данные не найдены

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была создана программа, обрабатывающая информацию о сдаче экзаменов в университет. Программа выполняет следующие функции:

* позволяет ввести информацию об абитуриентах;
* все введенные данные записываются в текстовый файл, что позволяет сохранить их и после закрытия программы;
* при необходимости можно изменить или удалить информацию об абитуриентах;
* программа позволяет просмотреть списки зачисленных студентов, как на определенные специальности, так и весь список зачисленных студентов на первый курс университета;
* программа осуществляет поиск абитуриента по введенной пользователем фамилии.

Программа реализована на алгоритмическом языке С++. В основе реализации поставленной задачи использовалась такая структура данных, как двусвязный список. Для реализации задачи и построения программы был реализован ряд функций, обеспечивающий функционирование и реализацию меню.

**СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ**

1. Адитья Бхаргава Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. 2017- 290с.
2. Роберт Седжвик Фундаментальные алгоритмы на С++ части 1- 4 Анализ структуры данных сортировка поиск 2001 - 688с.
3. Е.А.Аксенова, А.В.Соколов Алгоритмы и структуры данных на С++ Учебное пособие Петрозаводск издательство ПетрГУ 2008 - 82с.
4. Элджер, Джефф. C++: библиотека программиста / Дж. Элджер ; пер. с англ. Е. Матвеев. - Санкт-Петербург : Питер, 1999. - 320 с. : ил. - (Б-ка программиста)
5. Лафоре, Роберт Объектно-ориентированное программирование в C++ / Роберт Лафоре ; пер.: А. Кузнецов, М. Назаров, В. Шрага. - 4-е изд. - Санкт-Петербург [и др.] : Питер, 2017. - 923 с. : ил., схемы. - (Классика Computer science)

**ПРИЛОЖЕНИЕ А**

(обязательное)

Блок-схемы алгоритмов

начало

Пока не окончена работа

Menu()

Print ()

AddObj ()

**2** **1**

ChangeObj ()

ReadingObj ()

**3** **4**

DelObj ()

Sorting\_abcV()

1

Filter (1)

1

Sorting\_abcU()

Filter (2)

**5** 2

Sorting\_dv ()

2 **6**

Filter (3)

3

Sorting\_du ()

SavingObj ()

**7**

Sorting\_sv ()

3

FilterSt (1)

Sorting\_su ()

SearchObj ()

FilterSt (2)

**9**

**0** **8**

FilterSt (3)

FilterSt (4)

конец

Неверный выбор

**Рисунок А.1** Схема алгоритма программы

Загрузка из файла

Проверка даты

По возрастанию

Печать

Поиск

Сортировка даты

Подсчет количества элементов

Сохранение в файл

Добавление

Проверка текста

Проверка на ввод номера

Редактирование

Удаление

Сортировка

Меню

По убыванию

Печать записи

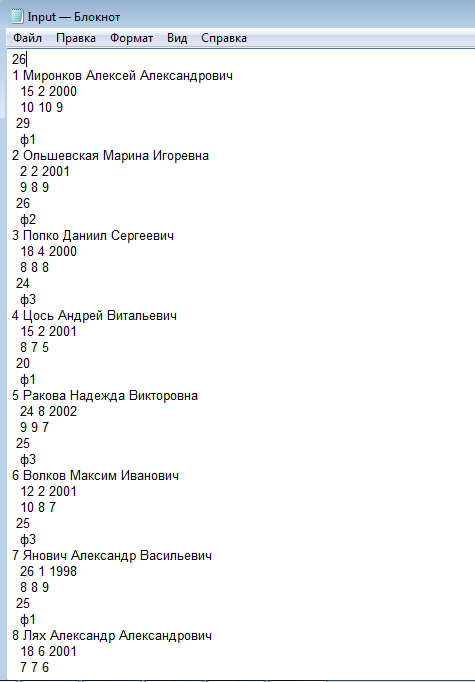
Печать таблицы

**Рисунок A.2** – Функциональная схема программы

**ПРИЛОЖЕНИЕ Б**

(обязательное)

Интерфейс программы



**Рисунок Б.1** вид текстового файла